VOCABULARY-INDEPENDENT SEARCH IN SPONTANEOUS SPEECH

Frank Seide, Peng Yu, Chengyuan Ma, and Eric Chang

Microsoft Research Asia, 5F Beijing Sigma Center, No. 49 Zhichun Rd., 100080 Beijing, P.R.C.

{fseide,t-rogery,i-chenma,echang}@microsoft.com

ABSTRACT

For efficient organization of speech recordings – meetings, interviews, voice mails, lectures – the ability to search for spoken keywords is an essential capability. Today, most spoken-document retrieval systems use large-vocabulary recognition. For the above scenarios, such systems suffer from both the unpredictable vocabulary/domain and generally high word-error rates (WER).

In this paper, we present a *vocabulary-independent* system to index and rapidly search spontaneous speech. A speech recognizer generates lattices of phonetic word fragments, against which keywords are matched phonetically.

We will first show the need to *use recognition alternatives* (lattices) in a high-WER context, on a word-based baseline. Then we will introduce our new method of *phonetic word-fragment lattice generation*, which uses longer-span language knowledge than a phoneme recognizer. Last we will introduce heuristics to *compact the lattices* to feasible sizes that can be searched efficiently.

On the LDC Voicemail corpus, we show that vocabulary/domain-*in*dependent phonetic search is as accurate as a vocabulary/domain-*dependent word-lattice* based baseline system for *in-vocabulary keywords* (FOMs of 74-75%), but nearly maintains this accuracy *also for OOV keywords*.

1. INTRODUCTION

Several approaches are reported in literature for the problem of searching for spoken words in audio recordings.

The TREC (Text REtrieval Conference) Spoken-Document Retrieval (SDR) track has fostered research on audio-retrieval of broadcast news clips. Most TREC benchmarking systems use broadcast news recognizers to generate approximate transcripts, and apply text-based information retrieval-techniques to these. They achieve retrieval accuracy similar to using human reference transcripts, and ad-hoc retrieval for broadcast news is considered a "solved problem" [1]. Noteworthy are the rather low word-error rates (20%), and that recognition errors did not lead to catastrophic failures due to redundancy of news segments and queries.

However, this approach does not address the problem of queries which are not in the recognizer's vocabulary. [2] reports that for the SpeechBot system, which indexes audio from public web sites, out-of-vocabulary (OOV) rates on the data are very low, but for the *queries* an OOV rate of 12% is observed. In [3] the same authors address the problem by indexing phonetic or phonetic word-fragment based transcriptions. For OOV queries, retrieval accuracy remains comparably low. Similar approaches, e.g. using overlapping M-grams of phonemes, are discussed in [4] and [5].

Vocabulary-independent indexing requires phoneme or word-fragment speech recognition. Due to low recognition accuracies, recognition alternatives must be considered in the search. [6] introduces the approach of searching phoneme lattices. [7] proposes a similar idea called "phonetic search track." Search is approached as a problem of *word-spotting*.

The system presented in this paper adopts the phoneme-lattice based word-spotting approach of [6]: "Indexing" consists of speech recognition to generate a phonetic representation of each audio file – a "lattice" of scored phoneme hypotheses – which can then be "searched" rapidly to locate keywords by its phonetic representation. Our system differs from [6] in that we use word-fragment based phoneme-lattice generation to make better use of prior language knowledge; a number of heuristics to improve lattice quality while reducing the data size and obliviating the need for confusion models; and the use of grammars to denote the query. To our knowledge, this is the first study that compares phonetic and word-lattice based search on a publicly available data set.

This paper is organized as follows. Section 2 describes the concept of lattice-based word spotting. Section 3 introduces our word-fragment approach to lattice generation. In section 4, we discuss efficient lattice-based search. Section 5 introduces our approaches to lattice compaction, and Section 6 the use of grammars. Sections 7 and 8 present the experimental setup and results.

2. LATTICE-BASED WORD SPOTTING

The result of a search operation is a list of hypotheses $(W, t_s, t_e, P(W\,t_s\,t_e|O))$ that match the query string W in the time range t_s to t_e. The posterior probability $P(W\,t_s\,t_e|O)$ serves as a measure for the goodness of match. In our phonetic approach, W is represented by its phoneme sequence (homophones become indistinguishable). O shall denote the acoustic observation expressed as a sequence of feature vectors o_t.

The posterior $P(W\,t_s\,t_e|O)$ is the sum of the probabilities of all paths that contain the query string W from t_s to t_e:

$$P(W\,t_s\,t_e|O) \;=\; \frac{\sum_{W_-,W_+} p(O\,t_s t_e|W_- W W_+) P(W_- W W_+)}{\sum_{W'} p(O|W')P(W')}$$

with W_- and W_+ denoting any word sequence before t_s and after t_e, respectively; W' being any word sequence; and[1]

$$p(O\,t_s t_e|W_- W W_+) = p(o_{0..t_s}|W_-)\,p(o_{t_s..t_e}|W)\,p(o_{t_e..T}|W_+)$$

$P(W\,t_s\,t_e|O)$ is efficiently approximated by forward-backward scoring of the word lattice. In a real implementation, acoustic scores must also be scaled by the inverse language-model weight [8].

[1] For crossword-triphone based recognizers, all terms becomes context dependent, and the above formulation needs to be extended accordingly.

3. PHONETIC WORD-FRAGMENT LATTICES

For vocabulary-independent phonetic search, the lattice does not contain words of a fixed vocabulary, but phonemes or phoneme sequences. Ideally, all valid phoneme sequences of the language should be allowed in the lattice (scored with an appropriate language-model probability), and invalid sequences excluded.

The simplest approach is to generate lattices using a phoneme dictionary and language model. However, longer units, such as syllables, are desirable to provide for better suppression of invalid phoneme sequences, more effective lattice pruning, and more accurate time boundaries from our speech recognizer, which uses the common word-pair approximation for lattice generation [9, 10].

We decided against the English syllables as units since we feared that due to their very large number (over 10,000), a syllable language model might not distinguish robustly between invalid and unseen phoneme sequences. Instead, like [3] we use an automatic method to generate a smaller set of units – *phonetic word fragments* – that is approximately optimal w.r.t. training-set perplexity.

3.1. Automatic Word-Fragment Selection

Klakow's iterative corpus mapping algorithm [11] based on mutual-information seemed the most suitable approach. With the mutual information of two units v and w defined as

$$\begin{aligned} \mathrm{MI}(v,w) &= \log\left(\frac{P(v,w)}{P(v)\cdot P(w)}\right)\cdot P(v,w) \\ &= \log\left(N_{\mathrm{tok}}\cdot\frac{N_{v,w}}{N_v\cdot N_w}\right)\cdot\frac{N_{v,w}}{N_{\mathrm{tok}}} \end{aligned}$$

an iteration will begin by determining all pairs (v,w) with MI above a threshold. Ambiguity from potentially overlapping pairs is resolved by eliminating a pair from the list if one or both constituents are part of a pair with higher MI. Pairs spanning word boundaries are also excluded. The remaining pairs $v\ w$ are replaced in the training corpus by single units v-w. The process is repeated until the desired number of fragments has been obtained.

Examples of fragments generated this way are /-k-ih-ng/ (the syllable *-king*), /ih-n-t-ax-r-/ (*inter-*), and /ih-z/ (the word *is*). One of the longest fragments turned out to be /ae-k-ch-uw-ax-l-iy/ (the word *actually*). The resulting mapped corpus also included a residual number of occurences of each individual phoneme.

3.2. Phoneme-Level Time Boundaries and Scores

An accurate time boundary should be determined for a keyword even if the keyword boundary falls into the middle of a multi-phone fragment. The speech recognizer's output needs to include time boundaries and scores of each individual phoneme.

Assuming a bigram fragment-language model, existing bigram decoders can be configured to implement this by replacing each fragment by a sequence of *fragment-dependent phonemes*, and converting the fragment bigram model into a special "equivalent" bigram model of fragment-dependent phonemes. Such an equivalent model cannot be guaranteed to be normalized locally, nor to provide accurate language-model probabilities for each *individual* phoneme. On the whole-sentence level, however, exact and normalized probabilities are possible.

4. EFFICIENT LATTICE SEARCH

In phonetic search, inverse-indexing techniques common in text-based information retrieval systems are infeasible due to the large amount of alternative recognition results. Nearly every lattice would match every query, even when using longer units such as phone-triplets. Full search of all lattices is unavoidable, and the time for searching one lattice a critical factor. Our search consists of two steps:

1. identify all lattice sub-paths that match the query string and compute their posterior probability;
2. merge matches with identical or nearly identical time boundaries.

The solution to step 1 is symmetric dynamic programming [6], which we implement by token passing. Each lattice node has an associated token stack. For each node n, beginning with the lattice start node, we (1) start a new hypothesis by inserting a new token into the nodes' stack (initialized with the node's time $t_s = t_n$ and the forward probability[2] $\alpha_n = \sum_W p(o_{0..t_n} W)$ and (2) try to expand all hypotheses in the token stack with their respective next phoneme in the query string by searching for matching outgoing edges. For expanded hypotheses, a new token is inserted into the edge's target node's token stack after accumulating the edge's acoustic and language model score. If the end of the query string is reached, the backward score at the end node is added, and the result is inserted into the result list.

A dramatic speed-up is achieved by integrating step 2: Tokens are only inserted into a token stack if this stack does not already contain a token that corresponds to the same phoneme position in the query string. If such a token is found (possible due to a different crossword context or different start time at the word beginning), both tokens are instead merged, their partial path probabilities added up. If the two tokens' start times differ, the more probable start time will propagate.

5. HEURISTICS FOR LATTICE COMPACTION

The resulting fragment-based phoneme lattices are infeasibly huge (av. 250 hypotheses/cs; requiring several 100 MB/h), mostly due to the expansion of triphone contexts. Yet they suffer from low hit rates, because often, triphone hypotheses to match a keyword are present but located on disconnected sub-paths of different fragments hypotheses, although they could form an acoustically valid path. To increase hit rates we apply the following post-processing:

1. Convert the lattice into a true triphone lattice to allow cross-over between fragments. Maintaining the triphone-context conditions keeps acoustic path scores accurate. However, fragment-based language-model scores become invalid and need to be replaced by a phoneme-level M-gram model.

2. Add penalized back-off paths to allow transitions between phoneme hypotheses with *mismatching* triphone context. Path scores may now consist of slightly inconsistent hypotheses.

Second, to bring memory requirements into a feasible range:

3. Eliminate all triphone-context constraints and merge triphone hypotheses to form a monophone lattice. "Estimate" the emission probabilities for the resulting monophone hypotheses by the very crude but effective heuristics of linearly combining the triphones' probabilities weighted by their posterior probability.

4. Time collapsing. Three frames are collapsed onto a single node. Scores for hypotheses whose length changes are renormalized (another crude heuristics).

The crude manipulations of the last two steps trade increased hit rates for distorted acoustic scores. As we will show below, this has different effect for long and short keywords.

[2] Again we ignore triphone context here for simplicity of presentation.

6. USING A GRAMMAR

We extended the lattice-search algorithm described in section 4 to accept the query-string in the form of a weighted finite-state network representing a grammar. This opens interesting possibilities.

1. Alternative pronunciations of a keyword can be searched for simultaneously. The probabilities for all variants will automatically be summed up, and it is easy to include prior probabilities for pronunciation variants.

2. For batch evaluations such as ours where sets of over 2000 different keywords need to be searched for, a huge speed-up can be achieved by folding the keyword list into a prefix tree.

3. The network allows generalized searches for complex expressions like telephone numbers or date expressions.

In this paper, we use the first two.

7. EXPERIMENTAL SETUP

7.1. Test database

We evaluate our system on the LDC Voicemail Corpus [12], which consists of two parts, VM-I and VM-II. The acoustic model was trained on 309 hours of the Switchboard corpus, and mixture means were MAP-adapted using the 15h training portion of VM-I.

The evaluation set is `vmtest` as defined in [13], consisting of the original test sets of VM-I and VM-II, plus additional material from VM-II training (we did not use VM-II for any training) – a total of 243 voice messages, 94 minutes of data. During algorithm development and for all parameter tuning, we used a separate development set consisting of 250 different VM-II messages.

To be truly vocabulary and domain independent, we did *not use voicemail data for phonetic search w.r.t. vocabulary, language model, or word-fragment generation*. Instead, we used phonetic transcriptions of the 309 hours of Switchboard, of 50 hours of LDC Broadcast News 96 training, and from 87,000 background-dictionary entries, total 11.8 million phoneme tokens.

For our *word-based baselines only*, we trained a language model on the VM-I training transcriptions[3] (about 160,000 words). The recognition lexicon is the training-set vocabulary of 7469 words.

7.2. Keyword selection

The keyword set to be used in our experiments was selected by an automatic procedure from the transcription of the test files.

We first picked all words and word sequences up to 4 words that appear in at most two different voice messages (document frequency ≤ 2), assuming they represent the most informative words to describe a document. This list was further filtered according to criteria similar to [7]. Stopwords[4] were excluded, as were numbers, genitives, spelled letters, words with 3 letters or less, and words (or sequences) that are substrings of other words (or of sequences of the same number of words). Stopwords occuring in at most 30 documents were allowed to be part of word sequences.

The resulting keyword set has 2049 entries, 620 of which (30.3%) are OOV[5] w.r.t. our word-based baseline. Example keywords are *pentium*, *federal-express package*, and *internet-workstation address request*.

[3] Interpolation with a Switchboard language model did not lead to improvements.

[4] The stoplist was a slightly extended version of the commonly used stoplist of the SMART information-retrieval system.

[5] A word sequence is out of vocabulary if at least one of its words is.

Table 1. Test-set perplexities (PP) for phoneme, fragment, and word language model.

	phoneme trigram (SWBD+)	fragment bigram (SWBD+)	word trigram (VM-I)
token PP	83	351	116
phoneme PP	83	43	4.7

7.3. Phoneme-lattice generation

A fragment vocabulary of 500 fragments (plus the original entire phoneme list) was created with the word-fragment generation procedure described in section 3. The 11.8 million phoneme tokens in the language-model training corpus were mapped to 6.2 million fragments, from which a bigram language model was trained.

Table 1 compares the development test-set perplexity of the fragment-based model with a phoneme trigram and the baseline word-level trigram model. We need to mention here that in our system, phonemes carry a word-position marker (word beginning, middle, end, or single-phone word), increasing the phoneme inventory to 166 units. Also note that the word trigram were trained on voice messages ("VM-I"), while phoneme and fragment models was trained on non-voicemail data (denoted as "SWBD+").

For phoneme-lattice generation, we used the HTK tool `HVite`, a general Viterbi decoder that can be configured to perform all sorts of decoding tasks by specifying the corresponding recognition network. It has direct support for crossword triphones and lattice generation, but no large-vocabulary recognition techniques such as M-gram language models or the use of a prefix tree.

We configured `HVite` with a static phoneme network representing the vocabulary as a prefix-tree structure that has been fully expanded w.r.t. language-model context (utilizing the language model's back-off nature) and contains fully factored bigram-language model scores (aka language-model lookahead). Because `HVite` considers every arc in the network a "word," the generated lattices contain the full phoneme-level segmentation and scoring while fully utilizing the longer language context provided by the word-fragment model.

8. RESULTS

We measure word-spotting accuracy by the common "Figure of Merit" (FOM) defined by NIST (National Institute of Standards & Technology) as the average of the detection/false-alarm curve over the range [0..10] false alarms per hour per keyword.

8.1. Word-based search (baseline) and top-1 vs. lattice search

First we present results from our *word-level baseline* system (Table 2). OOV keywords can of course not be found in the word-based output, so we excluded them here.

With the top-1 word-error rate (WER) of 40.2%, *word-spotting on the top-1 path* achieves a hit rate of 47.4%. The hit rate is an upper bound for the figure of merit (FOM) and is in turn bounded by the top-1 "correct rate"[6] which is 67.2% for our system.

Word-spotting on lattices yields a dramatic 60% relative improvement of FOM, to 75.2%. This is an important outcome to keep in mind when considering literature that benchmarks phoneme-*lattice* based search against *top-1* word-based search without reporting word-lattice based results (e.g. [7]).

[6] correct rate = 100% - (WER - insertion rate)

Table 2. Word-spotting results using the top-1 path vs. using word lattices (word-level baseline, in-vocabulary keywords only).

	top-1 path	lattice
WER	40.2%	-
FOM	$\leq$47.4%	75.2%

Table 3. Word-spotting results for phonetic search vs. word-based search (lattice).

	FOM	
keyword set	phonetic	word-based
in-vocabulary only	73.9%	75.2%
out-of-vocabulary only	70.1%	0
in- and out-of-vocabulary	72.8%	54.2%

8.2. Phonetic search and out-of-vocabulary keywords

Table 3 compares phonetic search and word-based search. For in-vocabulary keywords (w.r.t. the baseline vocabulary), phonetic search (FOM 73.9%) nearly reaches word-based search[7] (75.2%, a relative gap of $<$2%). Moreover, for phonetic search, the difference between in- and out-of-vocabulary keywords is small (relative gap of about 5%).

The overall FOM of phonetic search (72.8%) – including OOV keywords – is 34% higher that that of word-based search (54.2%).[8]

8.3. Effect of lattice-compaction heuristics

Table 4 shows the effect of the various lattice-compaction heuristics introduced in section 5. FOMs are given for the whole keyword set and separately for long keywords (7 phones or longer) and short keywords (2-6 phones). The table also shows average lattice sizes in edges/centisecond.

Heuristics 1. – conversion into true triphone lattice – leads to a significant FOM improvement (from 36.5% to 49.4%), caused by a similar increase of hit rate. The loss of long-span LM context has no visible impact. *Heuristics 2* – the addition of penalized back-off paths – brings a huge FOM improvement to 70.9%, again caused by a similarly large hit-rate increase. The mild inaccuracies of acoustic scores do not have visible impact.

Heuristics 3. is the elimination of triphone contexts and crude approximation of monophone emission probabilities. Here, the actual hit rates are instructive as well. For long words, the FOM now increases to 82%, at a hit rate of 90%. For short words, the FOM drops to 59.9% – at a hit rate of 94%. For *Heuristics 4.* – time quantization – this trend continues, but we now also see a small drop for the long words (at yet minimally increased hit rate).

Clearly, for long keywords, confusability is low, and hit rate is the limiting factor – the more phones in a keyword, the higher the chance that at least one of them has been pruned from the lattice. Short words on the other hand are highly confusable, so with hit rates approaching 100%, acoustic score is the dominating factor.

Luckily, long words occur less frequent, thus are more descriptive and more useful as keywords (our test set actually contains 40% more long keywords than short keywords). With all four heuristics applied, the resulting database of lattices becomes small enough to be searched within seconds.

[7] Please also note that the word-based baseline system's language model is biased towards voice messages, while the phoneme-based one is not.

[8] Of course, the actual OOV rate is the key factor here. Had we used a larger dictionary of, say 100,000 words, the gap might be smaller.

Table 4. Word-spotting results for different lattice-compaction heuristics, for all, long, and short keywords.

compaction	FOM			size
heuristics	all kw	long kw	short kw	[edges/cs]
baseline	36.5%	29.0%	47.0%	140
1. triphone	49.4%	44.4%	56.4%	122
2. back-off paths	70.9%	76.2%	63.3%	250
3. monophone	72.8%	82.0%	59.9%	30
4. time quant.	70.2%	81.3%	54.6%	10

9. CONCLUSION

We have presented a vocabulary-independent system to index and rapidly search spontaneous speech.

We have shown a substantial improvement from using lattices (unlike most SDR systems); presented a novel phonetic word-fragment based lattice-generation approach; and introduced heuristics for lattice compaction necessary to reduce storage size and to allow efficient search.

We found that vocabulary/domain-*in*dependent phonetic search is as accurate as a vocabulary/domain-*dependent* word-lattice based baseline system for in-vocabulary keywords, and nearly maintains this accuracy also for OOV keywords.

10. ACKNOWLEDGEMENTS

The authors wish to thank Dr. Asela Gunawardana for sharing his acoustic models for Switchboard. We also thank Dr. Mark Gales of Cambridge University for sharing his `vmtest` set definition.

11. REFERENCES

[1] J. Garofolo. TREC-9 Spoken Document Retrieval Track. National Institute of Standards and Technology, `http://trec.nist.gov/pubs/trec9/sdrt9_slides/sld001.htm`

[2] Beth Logan et al. An experimental study of an audio indexing system for the web. *Proc. ICSLP'2000*, Beijing, China, 2000.

[3] Beth Logan et al. Word and subword indexing approaches for reducing the effects of OOV queries on spoken audio. *Proc. HLT'2002*.

[4] P. Schäuble et al. First experiences with a system for content based rtrieval of information from speech recordings. *Proc. IJCAI'95*.

[5] Kenney Ng. Subword-based approaches for spoken document retrieval. PhD thesis, Massachusetts Institute of Technology, 2000.

[6] D. A. James and S. J. Young, A fast lattice-based approach to vocabulary-independent wordspotting. *Proc. ICASSP'04*, Adelaide, 1994.

[7] Mark Clements et al. Phonetic Searching vs. LVCSR: How to find what you really want in audio archives. AVIOS 2001.

[8] G. Evermann et al. Large Vocabulary Decoding and Confidence Estimation using Word Posterior Probabilities. *Proc. ICASSP'2000*, pp. 2366–2369, Istanbul, 2000.

[9] R. Schwartz et al. A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses. *Proc. ICSLP'94*, Yokohama, 1994.

[10] S. Ortmanns, H. Ney, F. Seide, and I. Lindam. A comparison of the time conditioned and word conditioned search techniques for large-vocabulary speech recognition. *Proc. ICSLP'96*, Vol. 2, pp. 1017-1020, Philadelphia, 1996.

[11] D. Klakow. Language-model optimization by mapping of corpora. *Proc. ICASSP'98*.

[12] M. Padmanabhan et al. Voicemail Corpus Part I (LDC98S77) and Part II (LDC2002S35). Linguistic Data Consortium, `http://www.ldc.upenn.edu`

[13] M. Gales et al. Porting: Switchboard to the Voicemail task. *Proc. ICASSP'03*, Hongkong, 2003.